Automatic generation of sources lemmas in Tamarin

Jannik Dreier

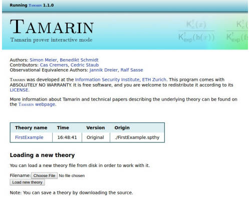
joint work with Véronique Cortier and Stéphanie Delaune

GDR Winter School
The Internet – February 10, 2021

Interaction and Automation

Tamarin's **interactive mode** allows the user to inspect and direct proof search

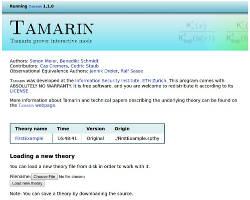
- Gives the **flexibility** required for complex case-studies
- Enables **fine-tuning** of models and proof strategies



On the downside, Tamarin's **automatic mode** often fails (compared to, e.g., ProVerif), even on relatively **simple examples**.

Tamarin's **interactive mode** allows the user to inspect and direct proof search

- Gives the **flexibility** required for complex case-studies
- Enables **fine-tuning** of models and proof strategies



On the downside, Tamarin's **automatic mode** often fails (compared to, e.g., ProVerif), even on relatively **simple examples**.

One of the **main reasons**: **partial deconstructions**.

Our **contribution**: **automatic handling of partial deconstructions** in most cases.

Plan

Plan

Consider the following toy protocol between the **initiator** and the **responder**:

1. $\rightarrow$: $\{\mathsf{req}, I, n\}_{\mathsf{pk}(R)}$

2. $\rightarrow$: $\{\mathsf{rep}, n\}_{\mathsf{pk}(I)}$

Toy example

Consider the following toy protocol between the **initiator** and the **responder**:

1. → : $\{\mathrm{req}, I, n\}_{\mathrm{pk}(R)}$
2. → : $\{\mathrm{rep}, n\}_{\mathrm{pk}(I)}$

In TAMARIN the initiator can be modeled using the following **rule**:

```
rule Rule_I:
    [ Fr(n),
      !Pk(R, pkR),
      !Ltk(I, ltkI) ]
  --[ SecretI(I, R, n) ]->
    [ Out(aenc{'req', I, n}pkR) ]
```

Toy example (Cont'd)

Consider the following toy protocol between the **initiator** and the **responder**:

1. → : $\{\mathsf{req}, I, n\}_{\mathsf{pk}(R)}$

2. → : $\{\mathsf{rep}, n\}_{\mathsf{pk}(I)}$

The responder can be modeled using the following **rule**:

```
rule Rule_R:
 [ In(aenc{'req', I, x}pk(ltkR)),
   !Ltk(R, ltkR),
   !Pk(I, pkI) ]
 --[ ]->
 [ Out(aenc{'rep', x}pkI) ]
```

Toy example (Cont'd)

Consider the following toy protocol between the **initiator** and the **responder**:

1. → : $\{\mathrm{req}, I, n\}_{\mathrm{pk}(R)}$

2. → : $\{\mathrm{rep}, n\}_{\mathrm{pk}(I)}$

Secrecy for the nonce n can be modeled using the following **lemma**:

```
lemma nonce_secrecy:
  "not(
      Ex A B s #i. SecretI(A, B, s) @ i
            & (Ex #j. K(s) @ j)
   )"
```

Toy example (Cont'd)

Consider the following toy protocol between the **initiator** and the **responder**:

1. $\rightarrow$: $\{\mathsf{req}, I, n\}_{\mathsf{pk}(R)}$
2. $\rightarrow$: $\{\mathsf{rep}, n\}_{\mathsf{pk}(I)}$

Secrecy for the nonce n can be modeled using the following **lemma**:

```
lemma nonce_secrecy:
  "not(
      Ex A B s #i. SecretI(A, B, s) @ i
            & (Ex #j. K(s) @ j)
   )"
```

Unfortunately, the **proof** of this lemma **does not terminate** due to partial deconstructions.

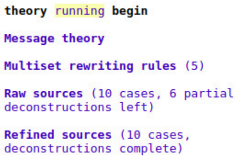
TAMARIN **pre-computes** all possible origins (called **sources**) of all protocol and intruder facts.

This can stop in an incomplete stage (called **partial deconstruction**) if TAMARIN lacks sufficient information about the origins of some fact(s).

```
theory running begin

Message theory

Multiset rewriting rules (5)

Raw sources (10 cases, 6 partial
deconstructions left)

Refined sources (10 cases,
deconstructions complete)
```

TAMARIN **pre-computes** all possible origins (called **sources**) of all protocol and intruder facts.

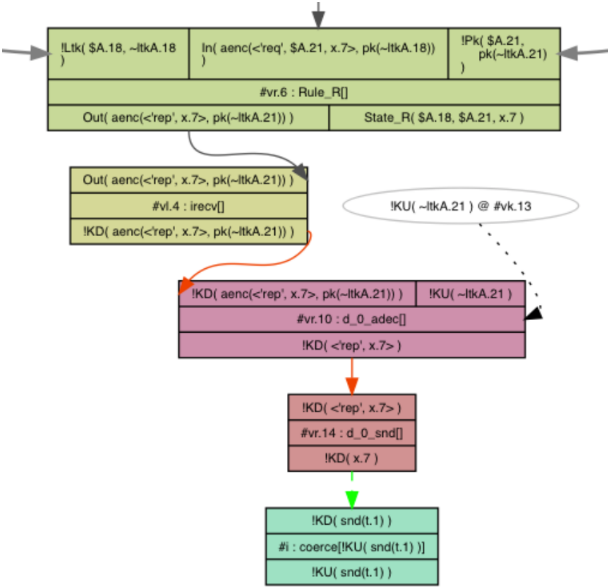
This can stop in an incomplete stage (called **partial deconstruction**) if TAMARIN lacks sufficient information about the origins of some fact(s).



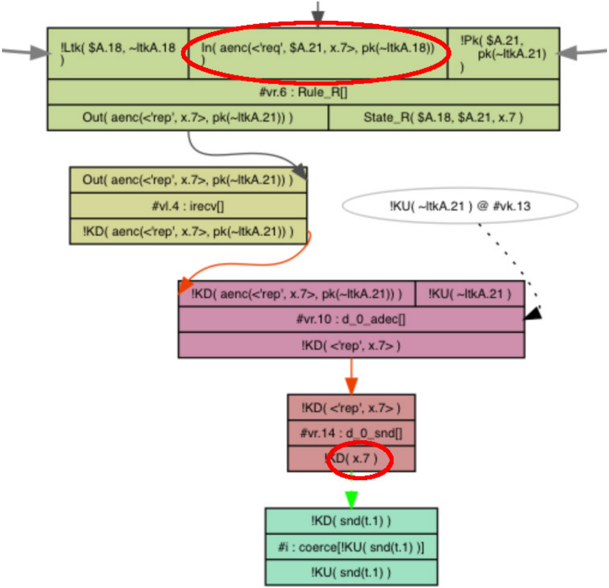
To **resolve** these partial deconstructions, one has to write a **sources lemma** detailing the possible origins of the problematic fact(s).

Sources lemmas are used to **refine** the sources, but they also need to be **proven correct**.

Example: Partial deconstruction



Example: Partial deconstruction



Example: Source lemma

We **know** that the input is either the message sent by the initiator, or a message constructed by the intruder.

We **know** that the input is either the message sent by the initiator, or a message constructed by the intruder.

Need to **annotate** the protocol rules:

```
rule Rule_I:
    [ Fr(n), !Pk(R, pkR),!Ltk(I, ltkI)]
  --[ I(aenc{'req', I, n}pkR), SecretI(I, R, n) ]->
    [ Out(aenc{'req', I, n}pkR) ]

rule Rule_R:
 [ In(aenc{'req', I, x}pk(ltkR)),
   !Ltk(R, ltkR), !Pk(I, pkI) ]
 --[ R(aenc{'req', I, x}pk(ltkR), x) ]->
 [ Out(aenc{'rep', x}pkI) ]
```

We **know** that the input is either the message sent by the initiator, or a message constructed by the intruder.

Need to **annotate** the protocol rules:

```
rule Rule_I:
    [ Fr(n), !Pk(R, pkR),!Ltk(I, ltkI)]
  --[ I(aenc{'req', I, n}pkR), SecretI(I, R, n) ]->
    [ Out(aenc{'req', I, n}pkR) ]

rule Rule_R:
 [ In(aenc{'req', I, x}pk(ltkR)),
   !Ltk(R, ltkR), !Pk(I, pkI) ]
 --[ R(aenc{'req', I, x}pk(ltkR), x) ]->
 [ Out(aenc{'rep', x}pkI) ]
```

Source lemma:

```
lemma typing [sources]:
"All x m #i. R(m,x)@#i ==> ((Ex #j. I(m)@#j  & #j < #i)
                          | (Ex #j. KU(x)@#j & #j < #i))"
```

Generalize idea & automate the approach:

1. Inspect the **raw sources** computed by TAMARIN
2. For each partial deconstruction:
    1. Identify the **variables** and **facts** causing the partial deconstruction
    2. Identify rules producing **matching conclusions**
    3. Add necessary **annotations** to the concerned rules
3. Generate a **sources lemma** using all annotations and add it to the theory

Generalize idea **& automate** the approach:

1. Inspect the **raw sources** computed by TAMARIN
2. For each partial deconstruction:
   1. Identify the **variables** and **facts** causing the partial deconstruction
   2. Identify rules producing **matching conclusions**
   3. Add necessary **annotations** to the concerned rules
3. Generate a **sources lemma** using all annotations and add it to the theory

Note that TAMARIN will **verify the correctness** of the generated lemma.

But we actually **proved** that the lemmas we generate are **correct** under some assumptions (well-formed rules, subterm-convergent equational theory).

How to identify matching conclusions?

First idea

Extract input message and try to **unify** with all outputs.

- Turns out to be **insufficient**, consider following example:
  - Input: $\langle \mathrm{enc}(a, k_1), \mathrm{enc}(b, k_2) \rangle$
  - Output 1: $\mathrm{enc}(a, k_1)$
  - Output 2: $\mathrm{enc}(b, k_2)$
  - **Unification fails**, but the intruder can easily compose both outputs

Solution

Use **protected subterms**:

- A protected subterm is subterm whose head symbol is **neither a pair** nor an **AC symbol**
- Allows us to abstract away pairs

Identifying matching conclusions

- Extract the **deepest** protected subterms **containing the variable** causing the partial deconstruction from the **facts** in the raw source

Example

$$t = \mathrm{enc}(\langle x, \mathrm{enc}(\langle b, x\rangle, k_2)\rangle, k_1)$$

has two deepest protected subterms w.r.t. x:

$$\mathrm{enc}(\langle b, x\rangle, k_2) \qquad \text{and} \qquad \mathrm{enc}(\langle x, \mathrm{enc}(\langle b, x\rangle, k_2)\rangle, k_1)$$

- Extract **all protected subterms** from all conclusions of all rules and try to **unify** with the deepest protected subterms
- If unification succeeds, we have a **match**.

Identifying matching conclusions

- Extract the **deepest** protected subterms **containing the variable** causing the partial deconstruction from the **facts** in the raw source

Example

$$t = \mathrm{enc}(\langle x, \mathrm{enc}(\langle b, x\rangle, k_2)\rangle, k_1)$$

has two deepest protected subterms w.r.t. x:

$$\mathrm{enc}(\langle b, x\rangle, k_2) \qquad \text{and} \qquad \mathrm{enc}(\langle x, \mathrm{enc}(\langle b, x\rangle, k_2)\rangle, k_1)$$

- Extract **all protected subterms** from all conclusions of all rules and try to **unify** with the deepest protected subterms
- If unification succeeds, we have a **match**.

Identifying matching conclusions

- Extract the **deepest** protected subterms **containing the variable** causing the partial deconstruction from the **facts** in the raw source

Example

$$t = \mathrm{enc}(\langle x, \mathrm{enc}(\langle b, x\rangle, k_2)\rangle, k_1)$$

has two deepest protected subterms w.r.t. x:

$$\mathrm{enc}(\langle b, x\rangle, k_2) \qquad \text{and} \qquad \mathrm{enc}(\langle x, \mathrm{enc}(\langle b, x\rangle, k_2)\rangle, k_1)$$

- Extract **all protected subterms** from all conclusions of all rules and try to **unify** with the deepest protected subterms
- If unification succeeds, we have a **match**.

We **implemented** the algorithm in TAMARIN (available in version 1.6.0).

To **enable** automatic source lemma generation, run TAMARIN with `--auto-sources`:

- If partial deconstructions are present and there is no sources lemma, the algorithm generates a lemma and adds it to the theory.
- If there is already a lemma, or there are no partial deconstructions, TAMARIN runs as usual.
- If a protocol rule has multiple variants, our algorithms considers all variants individually.

We tried numerous examples from the **SPORE library**:

Protocol Name	Partial Dec.	Resolved	Automatic	Time
Andrew Secure RPC	14	✓	✓	42.8s
Modified Andrew Secure RPC	21	✓	✓	134.3s
BAN Concrete Andrew Secure RPC	0	-	✓	10.6s
Lowe modified BAN Andrew Secure RPC	0	-	✓	29.8s
CCITT 1	0	-	✓	0.8s
CCITT 1c	0	-	✓	1.2s
CCITT 3	0	-	✓	186.1s
CCITT 3 BAN	0	-	✓	3.7s
Denning Sacco Secret Key	5	✓	✓	0.8s
Denning Sacco Secret Key - Lowe	6	✓	✓	2.7s
Needham Schroeder Secret Key	14	✓	✓	3.6s
Amended Needham Schroeder Secret Key	21	✓	✓	7.1s
Otway Rees	10	✓	✓	7.7s
SpliceAS	10	✓	✓	5.9s
SpliceAS 2	10	✓	✓	7.3s
SpliceAS 3	10	✓	✓	8.7s
Wide Mouthed Frog	5	✓	✓	0.6s
Wide Mouthed Frog Lowe	14	✓	✓	3.5s
WooLam Pi f	5	✓	✓	0.6s
Yahalom	15	✓	✓	3.1s
Yahalom - BAN	5	✓	✓	0.9s
Yahalom - Lowe	21	✓	✓	2.2s

We also tested all examples from the **Tamarin repository** that contained partial deconstructions:

Name	Partial Dec.	Resolved	Automatic	Time (new)	Time (previous)
Feldhofer (Equivalence)	5	✓	✓	3.8s	3.5s
NSLPK3	12	✓	✓	1.8s	1.8s
NSLPK3 untagged	12	✓	✗	-	-
NSPK3	12	✓	✓	2.4s	2.2s
JCS12 Typing Example	7	✓	✗	0.3s	0.2s
Minimal Typing Example	6	✓	✓	0.1s	0.1s
Simple RFID Protocol	24	✓	✗	0.7s	0.5s
StatVerif Security Device	12	✓	✓	0.3s	0.4s
Envelope Protocol	9	✓	✗	25.7s	25.3s
TPM Exclusive Secrets	9	✓	✗	1.8s	1.8s
NSL untagged (SAPIC)	18	✓	✓	4.3s	19.9s
StatVerif Left-Right (SAPIC)	18	✓	✓	28.8s	29.6s
TPM Envelope (Equivalence)	9	✗	-	-	-
5G AKA	240	✗	-	-	-
Alethea	30	✗	-	-	-
PKCS11-templates	68	✗	-	-	-
NSLPK3XOR	24	✗	-	-	-
Chaum Offline Anonymity	128	✗	-	-	-
FOO Eligibility	70	✗	-	-	-
Okamoto Eligibility	66	✗	-	-	-

- For **all** examples from SPORE, our approach was **successful** in resolving the partial deconstructions, and the entire verification became **automatic**.
- In **most** examples from the TAMARIN repository, our approach was also successful, including examples with **equivalence** properties or generated by **SAPIC**. Verification **times were similar** to manual source lemmas.
- In **some** cases the partial deconstructions were **resolved** but the rest was **not automatic**: further intermediate lemmas or other annotations were required
- Our approach **failed** for three reasons:
  - A too complex **equational theory** (not subterm convergent, AC symbols, . . .)
  - Partial deconstructions caused by **state facts** rather than messages
  - TAMARIN **fails to prove** the generated sources lemma

Conclusion & Future Work

- Automation in TAMARIN often fails because of **partial deconstructions**
- Developed & implemented a new algorithm to **automatically generate** sources lemmas
- Proved **correctness** of the generated lemmas
- Algorithm **works well in practice**, many examples become fully or at least partly **automatic**
- Available in TAMARIN 1.6.0
- **Future work:**
  - Handle more general **equational theories**
  - Handle partial deconstructions stemming from **state facts** (work in progess)